Network Working Group
Request for Comments #663
NIC #31387

Rajendra K. Kanodia
MIT, Project MAC
November 29, 1974

A LOST MESSAGE DETECTION AND RECOVERY PROTOCOL

1.0 INTRODUCTION

The current Host-to-Host protocol does not provide for the following three aspects of network communication:

1. detection of messages lost in the transmission path
2. detection of errors in the data
3. procedures for recovery in the event of lost messages or data errors.

In this memo we propose an extension to the Host-to-Host protocol that will allow detection of lost messages and an orderly recovery from this situation. If Host-to-Host protocol were to be amended to allow for detection of errors in the data, it is expected that the recovery procedures proposed here will apply.

With the present protocol, it may some times be possible to detect loss of messages in the transmission path. However, often a lost message (especially one on a control link) simply results in an inconsistent state of a network connection. One frequent (and frustrating) symptom of a message loss on a control link has been the "lost allocate" problem which results in a "paralyzed" connection. The NCP (Network Control Program) at the receiving site believes that sender has sufficient allocation for a connection, whereas the NCP of the sending host believes that it has no allocation (due to either loss of or error in a message that contained the allocate command). The result is that the sending site can not transmit any more messages over the connection. This problem was reported in the NWG-RFC[1] #467 by Burchfiel and Tomlinson. They also proposed an extension to the Host-to-Host protocol which allows for resynchronization of the connection status. Their proposed solution was opposed by Edwin Meyer (NWG-RFC #492) and Wayne Hathaway (NWG-RFC #512) on the grounds that it tended to mask the basic problem of loss of messages and they suggested that the fundamental problem of message loss should be solved rather than its symptoms. As an alternative to the solution proposed in NWG-RFC #467, Wayne Hathaway suggested that Host-to-Host protocol header could be extended to include a "Sequence Control Byte" to allow detection of lost messages. At about the same time Jon Postel suggested a similar scheme using message numbers (NWG-RFC #516). A little later David Walden proposed that four unused bits of the message sequence number (in the IMP leader) be utilized for sequencing messages (NWG-RFC #534). His scheme is similar to those proposed by Postel and Hathaway; however it has the advantage that Host-to-Host protocol mechanisms can be tied into the IMP-to-Host protocol mechanisms.

---

[1] (ARPA) Network Working Group - Request for Comments

The protocol extension proposed here uses the four bits of the message sequence number in the message leader for detection of lost messages. However, to facilitate recovery, it uses another eight bit field (presently unused) in the 72 bit header of the regular messages. In the next section of this paper we discuss some of the basic ideas underlying our protocol. In section 3, we provide a description of the protocol. It is our intention that section 3 be a self-contained and complete description of the protocol.

2.0 BASIC IDEAS

The purpose of this section is to provide a gentle introduction to the central ideas on which this protocol is based. Roughly speaking, our protocol can be divided into three major components. First is the mechanism for detecting loss of messages. Second is the exchange of information between the sender and the receiver in the event of a message loss. For reasons that will soon become obvious, we have termed this area as "Exchange of Control Messages". The third component of our protocol is the method of retransmission of lost messages. In this section, we have reversed the order of discussion for the second and third components, because the mechanisms for exchange of control messages depend heavily upon the retransmission methods.

A careful reader will find that several minor issues have been left unresolved in this section. He (or she) should remember that this section is not intended to be a complete description of the protocol. Hopefully, we have resolved most of these issues in the formal description of the protocol provided in the section 3.

2.1 DETECTION OF LOSS OF MESSAGES

The 32 bit Host-to-IMP and IMP-to-Host leaders contain a 12 bit message-id in bit positions 17 to 28 (BBN Report #1822). The Host-to-Host protocol (NIC[1] 8246) uses 8 bits of the message-id (bit positions 17 to 24) as a link number. The remaining 4 bits of the message-id (bits 25 to 28) are presently unused. For the purposes of the protocol to be presented here, we define these four bits to be the message sequence number (MSN in short) associated with the link. Thus message-id consists of an eight bit link number and a four bit message sequence number. The four bit MSN provides a sixteen element sequence number for each link. A network connection has a sending host (referred to as "sender" henceforth), a receiving host (referred to as receiver henceforth), and a link on which messages are transmitted. In our protocol the sender starts communication with the value of MSN set to one (i.e. the first message on any link has one in its MSN field.) For the next message on the same link the value of MSN is increased by one. When the value of MSN becomes 15 the next value chosen is one. This results in the following sequence 1, 2, ..., 13, 14, 15,

---

[1] (ARPA) Network Information Center

1, 2, ..., etc. The receiver can detect loss of messages by examining this sequence. Each hole corresponds to a lost message. Notice that the detection mechanism will fail if a sequence of exactly 15 messages were to be lost. For the time being, we shall assume that the probability of loosing a sequence of exactly 15 messages is negligible. However, we shall later provide a status exchange mechanism (Section 2.6) that can be used to prevent this failure.

Notice that in the sequence described above we have omitted the value zero. Following a suggestion made by Hathaway (NWG-RFC #512) and Walden (NWG-RFC #534) the new protocol uses the value zero to indicate to the receiving host that the sending host is not using message sequence numbers. We, in fact, extend the meaning associated with the MSN value zero to imply that the sending host has not implemented the detection and error recovery protocol being proposed here.

2.2 COMPATIBILITY

The discussion above brings us to the issue of compatibility between the present and the new protocols. Let us define the hosts with the present protocol to be type A and the hosts with the new protocol to be type B. We have three situations:

1. Type A communicating with type A: there is no difference from the present situation.
2. Type A communicating with type B: from the zero value MSNs in messages sent by the type A host, the type B host can detect the fact that the other host is a type A host. Therefore the type B host can simulate the behaviour of a type A host in its communication with the other host, and the type A host will not be confused. As we will see later that this simulation is really simple and can be easily applied selectively.
3. Type B host communicating with type B: Both hosts can detect the fact that the other host is a type B host and use the message detection and error recovery protocol.

There is one difficulty here that we have not yet resolved. When starting communication how does a type B host know whether the other host is type A or type B? This difficulty can be resolved by assuming that a type A host will not be confused by a non-zero MSN in the messages that it receives. This assumption is not unreasonable because a type A host can easily meet this requirement by making a very simple change to its NCP (the Network Control Program), if it does not already satisfy this requirement. Another assumption that is crucial to our protocol, is that the type A hosts always set the MSN field of messages (they send out) to zero. As of this writing, the author believes that no hosts are using the MSN field and therefore no compatibility problem should arise.

2.3 RETRANSMISSION OF MESSAGES

Before getting down to the details of the actual protocol, we will attempt here to explain the essential ideas underlying this protocol by considering a somewhat simplified situation. Consider a logical communication channel X, which has at its disposal an inexhaustible supply of physical communication channels C(1), C(2), C(3), ..., etc.[1] Channel X is to be used for transmission of messages. In addition to carrying the data, these messages contain (1) the channel name X, and (2) a Message Sequence Number (MSN). Let us denote the sender on this channel by S and the receiver by R. Let us also assume that at the start of the communication, R and S are synchronized such that R is prepared to receive messages for logical channel X on the physical channel C(1) and S is prepared for sending these messages on C(1). S starts by pumping a sequence of messages M(1), M(2), M(3), ..., M(n) into channel C(1). Since these messages contain sequence numbers, R is able to detect loss of messages on the channel C(1). Suppose now that R discovers that message number K (where K <n) was lost in the transmission path. Let us further assume that having discovered loss of a message, R can communicate this fact to S by sending an appropriate control message on another logical channel that is explicitly reserved for transmission of control messages from R to S. This channel, named Y, is assumed to be completely reliable.

We now provide a rather simplistic recovery protocol for the scenario sketched above. Having detected the loss of message M(K) on channel X, R takes the following series of actions:

1- R stops reading messages on C(1),
2- R discards those messages that were received on C1 and are placed after M(K) in the logical message sequence,
3- R prepares itself to read messages M(K), M(K+1), ..., etc. on the physical channel C(2),
and 4- R sends a control message to S on control channel Y, which will inform S to the effect that there was an error on logical channel X while using physical channel C(1) in message number K.

When S receives this control message on Y, it takes the following action:

1- S stops sending messages on C(1),
and 2- begins transmission of messages starting with the sequence number K, on the physical channel C(2).

---

[1] One method of recovery may be to let the receiver save all properly received messages and require the sender to retransmit only those messages that were lost. This method requires the receiver to have the ability to reassemble the messages to build the data stream. A second method of recovery may be to abort and restart the transmission at the error point. This method requires that the receiving host be able to distinguish between legitimate messages and messages to be ignored. For simplicity we have chosen the second method and an inexhaustible supply of physical channels serves to provide the distinction among messages.

This resynchronization protocol is executed every time R detects an error. If physical channel C(CN) was being used at the time of the error, then the next channel to be used is C(CN+1). We can define a "receiver synchronization state" for the channel X, as the triplet R(C, CN, MSN), where C is the name of the group of physical channels, CN is the number of the physical channel in use, and MSN is the number of message expected.[1] We can specify a message received on a given C-channel as M(MSN). When R receives the message M(R.MSN) on the channel C(R.CN), the synch-state changes from R(C, CN, MSN) to R(C, CN, MSN+1). However if M.MSN for the message received is greater than R.MSN then a message has been lost, and R changes the synch-state to R(C, CN+1, MSN). What really happens may be described as follows: upon detection of error in a logical channel X, we merely discard the physical channel that was in use at the time of error, and restart communication on a new physical channel at the point where break occurred.

This scheme provides a reliable transmission path X, even though the physical channels involved are unreliable. In this scheme we have assumed that (1) a completely reliable channel Y is available for exchange of control messages, and (2) that there is a large supply of physical channels available for use of X. In the paragraphs that follow we shall revise our protocol to use a single physical channel and then apply this protocol to the channel Y in such a way that Y would become "self-correcting."

Now suppose that channel X has only one physical channel (named X') available for its use rather than the inexhaustible supply of physical channels. Our protocol would still work, if we could somehow simulate the effect of a large number of C-channels using the single channel X'. One method of providing this simulation is to include in each message the name of the C-channel on which it is being sent, and send it on X'. Now the receiver must examine each message received on X' to determine the C-channel on which this message was sent. Our protocol still works except for one minor difference, namely, the receiver must now discard messages corresponding to C-channels that are no longer in use, whereas in the previous system the C-channels no longer being used were simply discarded. To be sure, X' can be multiplexed among only a finite number of C-channels; however, we can provide a sufficiently large number of C-channels so that during the life time of the logical channel X, the probability of exhausting the supply of C-channels would be very low. And even if we were to exhaust the supply of C-channels, we could recycle them just as we recycle the message sequence numbers.

A physical message received on X' can now be characterized by a pair of C-channel number and a message sequence number, as M(CN, MSN). The receiver synchronization state becomes a triplet R(X', CN, MSN). This state tells us that R is ready to receive a message for X on the physical channel X' and for this message M.CN should be equal to R.CN and M.MSN should be equal to R.MSN. All messages with M.CN less than R.CN will be ignored. If for the next message received on X', M.CN = R.CN and M.MSN = R.MSN, then R changes the synch state to R(X', CN, MSN+1). If M.CN = R.CN but M.MSN > R.MSN then a message has been lost and the synch-state R(X', CN, MSN) changes to R(X', CN+1, MSN). Notice that we have not yet said anything about the situation M.CN > R.CN. We will later describe a scheme for using this case to provide for error correction on the control channel itself.

---

[1] Notice that we have prefixed this triplet by the letter R (for the receiver.) We will prefix other similarly defined quantities by different letters. For example M can be used for messages. This notation permits us to write expressions like M.MSN = R.MSN, where M.MSN stands for the message sequence number of the message.

2.4 EXCHANGE OF CONTROL INFORMATION

So far we have discussed two schemes for the detection and retransmission aspects of the lost-message problem. In this section, we discuss methods by which the receiver communicates to the sender the fact of loss of messages.

We continue with the scenario developed in the above section with a small change. For the purposes of the discussion that is about to follow we shall assume that there are actually two perfect channels available for exchange of control messages. One channel from S to R named S->R, and the other from R to S named R->S. The purpose of S->R will become clear in a moment. In order to let R communicate the fact of loss of messages to S, We provide a control message called Lost Message from Receiver (LMR) which is of the following form: LMR(X, CN, MSN), where X is the name of the channel, CN is the new C-channel number, and MSN is the message sequence number of the lost message. If more than one message has been lost, then R uses the MSN of the first message only. When S receives this message, it can restart communication at the point where the break occurred using the C-channel specified by the LMR message. This will restore the communication path X. What happens if S can not restore communication at the break point because it does not have the relevant messages any more? This issue can be solved in one of the two ways: either let the protocol specify a fixed rule that S will be required to close the connection, or the protocol could allow S and R (and may be the users on whose behalf S and R are communicating on X) to negotiate the action to be taken. For the protocol to be presented here, we have taken the approach that S may, at its option, either close the connection or negotiate with R. What action a specific host takes can either be built into its NCP or determined dynamically. Those hosts that do not have very powerful machines will probably chose the static option of closing the connection; other hosts may make the decision depending upon the circumstances. For example, a host may decide that loss of messages is not acceptable during file transfers whereas loss of a single message can be ignored for terminal output to interactive users. A host might even let the user processes decide the course of action to be taken. If S determines that it can not retransmit lost messages, it may want to let R decide what action is to be taken. If S so decides, then it may communicate this fact to R by transmitting a Lost Message from Sender (LMS) control message to R on the channel S->R. An LMS message is of the following form: LMS(X, CN, MSN, COUNT), where X is the name of the channel, CN is the name of the C-channel obtained from the LMR message, MSN is the message sequence number of the first message in the sequence of lost messages, and COUNT is the number of messages in the sequence. S resets its own synch-state for connection X to S(X, CN, MSN+COUNT). When S has informed R of its inability to retransmit lost messages, the burden of the decision falls on R, and S simply awaits R's reply before taking any action for the channel X. When R receives the LMS, it may either decide that loss is unacceptable and close the connection, or it may decide to let S continue. In the later case R informs S of its decision to continue by transmitting a Loss of Message Acceptable (LMA) control message to S. Upon receiving the LMA control message, S resumes transmission on X. To avoid the possibility of errors in exchange of control messages, the LMA control message is specified to be an exact replica of the LMS control message, except for the message code which determines whether a control message is LMA or LMS or something else.

In general, a sending host has only a limited amount of memory available for storing messages for possible retransmission later. In section 2.6 we provide a status exchange mechanism that can be used to determine when to discard these messages.

2.5 RECOVERY ON CONTROL LINKS

We continue our discussion with the scenario developed in the previous section. The receiver R may detect loss of messages on control channels by examining the message sequence numbers on the messages. When R detects that a message has been lost on the channel S->R, it (R) may transmit an LMR to S on R->S communicating the fact of loss of messages. When S receives the LMR for the control link, it must either retransmit the lost messages, or "close" the connection. (In the context of Host-to-Host protocol, closing the network connection for control link implies exchange of reset commands, which has the effect of reinitializing all communication between R and S.) For control links, S does not have the option of sending an LMS to the receiver. If S can not retransmit the lost messages then it aborts the output queue (if any) for the channel S->R, and inserts a Reset command at the break point. Essentially, we are specifying that if S can not retransmit lost control messages then the error would be considered irrecoverable and fatal. All user communication between S and R is broken and must be restarted from the beginning.

In the above paragraph, we considered the situation in which R was able to detect the loss of messages. It is however possible that it is the last message transmitted on S->R that is lost. In this case, R will not be aware of the loss. In this situation, recovery can be initiated only if S can either positively determine or simply suspect that a message has been lost. In general, after having transmitted a control message, S would be expecting some sort of response from R. For example, if S transmits a Request-for-Connection (RFC) control message to R, S expects R to reply either with a Close (CLS) command or another RFC. If, after an appropriate time interval has elapsed and S has received no reply from R, our protocol specifies that S may retransmit the control message. In retransmitting, S must use the same C-channel and MSN values that were used for the original message. Since R can, now, receive duplicate copies, we stipulate that if R receives a duplicate of the message that it has already received, it may simply ignore the duplicate.

2.6 SOME OTHER PROBLEMS

There are two problems that have not yet been solved. First, a sending host will usually have only a limited amount of buffer space in which it can save messages for possible later retransmission. So far, we have not provided any method by which a host may positively determine whether the receiver has correctly received a certain message or not. Thus it has no firm basis on which it may decide to release the space used up by messages that have been already transmitted. The second problem is created by our recycling the message sequence numbers. For the MSN mechanism to work correctly, it is essential that at any given instant of time there be no more than 15 messages in the transmission path. If there were more than 15 messages, some of these messages would have same MSN and LRN, and if one of these messages were to be lost, it would be impossible to identify the lost message. However, the second problem should not arise in the ARPA Network, since the IMP sub-network will not allow more than eight outstanding messages between any host pair (NWG-RFC #660).

We can solve both these problems by a simple yet highly flexible scheme. In this scheme, there are two new control messages. One of these, called _Request Status from Sender_ (RSS), can be used by the sending host to query the receiver regarding the receiver's synch-state. The receiver can supply its copies of C-channel number and MSN for a transmission path by sending a _Status from Receiver_ (SFR) control message over the control channel. An SFR provides positive acknowledgement; differing with the usual acknowledgement schemes in only that here acknowledgement is provided only upon demand. Upon receiving SFR, the sender knows exactly which messages have been properly delivered, and it may free the buffer space used by these messages. The RSS and SFR scheme can also be used to ensure that there are no more than fifteen messages in the transmission path at any given time.

3.0 LOST MESSAGE DETECTION AND RECOVERY PROTOCOL

This protocol is proposed as an amendment to the Host-to-Host Protocol for the purpose of letting hosts detect the loss of messages in the network. It also provides recovery procedures from such losses. This protocol is divided into two parts. Part 1 states the compatibility requirements. Part 2 states the new protocol and must be implemented by hosts that desire to have the ability to recover from loss of messages in the network. The reader will find many comments interspersed throughout the description of this protocol. These comments are not part of the protocol and are provided solely for the purpose of improving the reader's understanding of this protocol.

The terminology used in this protocol is similar to that used in the Host-to-Host protocol. We speak of a "network connection" between a pair of hosts, called the "receiver" and the "sender." A network connection is described by a pair of socket numbers, and once established, a network connection is associated with a link (which is described by a link number.) A network connection is a logical communication path and the link assigned to it is a physical communication path. In addition to links associated with the network connections, there are "control-links" for the transmission of "control commands." When we use the term "connection" it may refer to either a network connection or the link assigned to it; the context decides which one. The term "links" encompasses the connection-associated-links as well as control-links. Notice that a receiver of a connection may transmit control commands regarding this connection.

3.1 DEFINTIONS

3.1.1 HOST TYPE "A"

Those hosts that have not adopted the part 2 of this protocol amendment will be known as type A hosts.

(Comment: All existing hosts are type A hosts.)

3.1.2 HOST TYPE "B"

Those hosts, that adopt the part 2 of this protocol amendment will be known as type B hosts.

3.1.3 MESSAGE SEQUENCE NUMBER (MSN)

A four bit number associated with regular messages and contained in the bits 25 through 28 of the Host-to-IMP and IMP-to-Host leaders for regular messages [BBN

Report No. 1822]. This number is used by the type B hosts to detect loss of messages on a given link. Type A hosts always set the MSN (for the messages they send out) to zero. When in use by a type B host, the first message on a link (after the connection has been established) has the MSN value of one. For each successive message on this link, the MSN value is increased by one until it reaches a value of 15. The next message has the MSN value of one.

(Comments: Type B hosts will use the MSN mechanism only when communicating with other type B hosts. If a type B host is communicating with a type A host, the type B host will essentially simulate the behaviour of a type A host and use zero MSN values for this communication.)

3.1.4 LINK RESYNCH NUMBER (LRN)

The Link Resynch Number is an eight bit number associated with a link and used for resynchronizing the communication. For links associated with a network connection (i.e. user links), it is intially set to zero. For control links, it is set to zero at the time of the Network Control Program (NCP) initialization. For a given link, its current LRN value is copied into the LRN field of all messages sent out on the link. The LRN values may be manipulated by type B hosts in accordance with the protocol described later.

(Comments: Our protocol specifies that for all communication involving a type A host, the LRN value will never change from zero. Since the LRN field is presently unused, all hosts set it to zero even though they do not explicitly recognize this field as an LRN field. This guarantees compatibility.)

3.1.5 LRN FIELD

An eight bit field in the bits 33 through 40 of the Host-to-Host message header.

3.2 NEW CONTROL COMMANDS

3.2.1 LMR - LOST MESSAGE FROM RECEIVER

```
    8       8       8       8
 _______________________________
|       I       I       I       I
I  LMR  |  link |  LRN  |  MSN  I
I_______I_______I_______I_______I
```

The LMR command is sent by a receiving host to let the sending host know that one or more messages have been lost. The MSN field specifies the message sequence number of the first message lost. The LRN field specifies the new LRN value that must be used if and when communication is restored.

(Comments: As will be seen later, the LMR command also has the effect of resetting the bit and message allocation in the sending host to zero. In order to permit a sender to restore communication, an LMR command will be usually accompanied with an allocate command. However notice that these comments do not apply to the control links because there is no allocation mechanism for the control links.)

3.2.2 LMS - LOST MESSAGE FROM SENDER

```
      8        8       8        8        8
 _____________________________________________
 I        I       I       I         I       I
 I  LMS   I Link  I  LRN  I  MSN    I COUNT I
 I________I_______I_______I_________I_______I
```

This command is sent by a sender host in reply to an LMR command if it (the sender) can not retransmit the lost messages specified by the LMR command. The purpose of this command is to query the receiver whether or not the loss of messages is acceptable. After sending this command, the sender waits for a reply before transmitting any messages over the link involved. This command may not be sent for control links. The LRN and MSN values are same as those specified by the LMR command. COUNT specifies the number of messages lost.

3.2.3 LMA - LOSS OF MESSAGES ACCEPTABLE

This command is identical to the LMS command accept for the command code. Upon receipt of an LMS command, a receiver may send this command to the sender to let the sender know that loss of messages is acceptable. All fields in this command are set to corresponding values in the LMS command.

3.2.4 CLS2 - CLOSE2

```
      8          32               32              8       8
 ____________________________________________________________
 I       I                I                 I       I       I
 I CLS2  I   my socket    I your socket     I  LRN  I MSN   I
 I_______I________________I_________________I_______I_______I
```

The CLS2 command is similar to the current CLS command except for the LRN and MSN fields included in the new command. Both the receiving and sending hosts copy their values of LRN and MSN into the CLS2 command. Upon receiving a CLS2 command, a host compares the LRN and MSN values contained in the CLS2 command with its own values for the connection involved. If these values do not match, then an error has occurred and a host may initiate recovery procedures.

(Comments: The purpose of this command is to ensure that the last message transmitted on a connection has been received succesfully.)

3.2.5 ECLS - ERROR CLOSE

```
      8          32            32
 _____________________________________
 I      I             I              I
 I ECLS I my socket   I  your socketI
 I______I_____________I______________I
```

The ECLS command is similar to the current CLS command. It is used for closing connections which have sufferred an irrecoverable loss of messages.

(Comments: A connection may be closed in any one of the following three ways:

1. A connection which has not yet been opened succesfully may be closed by the CLS command. All connections involving type A hosts must be closed using the CLS command.
2. Connections between type B hosts may be closed using CLS2 command. A connection is considered closed only if matching CLS2 commands have been exchanged between the sender and the receiver.
3. Those connections between type B hosts, that have sufferred an irrecoverable loss of messages, must be closed by the ECLS command.)

3.2.6 RSS - REQUEST STATUS FROM SENDER

```
     8      8
 _____________________
 I      I      I
 I RSS  I LINK I
 I______I______I
```

A sending host may issue an RSS command to find out about the status of transmission on a particular connection or the control link.

3.2.7 RSR - REQUEST STATUS FROM RECEIVER

```
 _____8________8______
 I        I          I
 I RSR    I LINK     |
 I________I__________I
```

A receiver can issue an RSR command to find out about the status of transmission on a particular connection or the control link.

3.2.8 SFR - STATUS FROM RECEIVER

```
 _____8________8________8________8_____
 I        I        I        I         I
 I SFR    I  LINK  I  LRN   I MSN     I
 I________I________I________I_________I
```

A receiving host may issue this command to inform the sender about the state of a particular connection or the control link.

3.2.9 SFS - STATUS FROM SENDER

```
 _____8________8________8________8_____
 I        I        I        I         I
 I SFS    I  LINK  I  LRN   I MSN     I
 I________I________I________I_________I
```

A sending host may issue this command to inform the receiver about the state of a particular connection or the control link.

3.3 THE PROTOCOL

3.3.1 PART ONE

All type A hosts must use zero MSN and LRN values on the messages sent out by them. When communicating with a host of type A, a type B host must simulate the behaviour of type A host.

(Comments: Notice that this simulation is not complicated at all. All that is required is that hosts that adopt this protocol must not use it when communicating with the hosts that have not adopted it.)

3.3.2 PART TWO

This part of the protocol is stated as a set of rules which must be observed by all type B hosts when communicating with other type B hosts.

3.3.2.1 RESPONSIBILITIES OF HOSTS AS SENDERS

(1). A type B sending host must use message sequence numbers on all regular messages that it sends to other type B hosts as specified in the definition of the message sequence numbers (Section 3.1.3).

(2). A type B sending host must use link resynch numbers on all regular messages that it sends to other type B hosts as specified in the definition of link resynch number (Section 3.1.4).

(3). A sending host may retransmit a message if it suspects that the message may have been lost in the network during previous transmission.

(4). A sending host may issue an RSS command to the receiver to determine the state of transmission on any link.

(5). A sending host must use the ECLS command to close a connection, if the connection is being closed due to an irrecoverable transmission error. Otherwise, it must the CLS2 command.

3.3.2.2 RESPONSIBILITIES OF HOSTS AS RECEIVERS

(1). A receiver host will maintain LRN and MSN values for each link on which it receives messages. Initial value of LRN will be zero, and initial value of MSN will be one. For each receive link, the host should be prepared to receive a message with LRN and MSN values specified by its tables. When the host has received the expected message on a given link, it will change its table MSN value as specified in the definition of MSN.

(2). On a given link, if a host receives a message with an LRN value smaller than the one in use, it will ignore the message.

(3). If a host receives a duplicate message (same LRN and MSN values), it will ignore the duplicate.

(4). A host will examine the MSN values on all regular messages that it receives to detect loss of messages. If on any link, one or more messages are found missing, it will concern itself with only the first message lost and take the following series of action:

1. Increase its own LRN value for this link by one.
2. Send an LMR command to the sending host with LRN field set

to the new value and MSN field set to the sequence number of the first message lost.
3. Realizing that LMR command will cause the allocation to be reset to zero, it will send more allocation. This is not applicable to the control links.

However, if a host does not want to initiate the recovery procedures, it may simply close the connection by an ECLS command.

(5). A receiver host may issue the RSR command to determine the state of transmission on any link.

(6). If a connection is being closed due to an irrecoverable error, a receiving host must use the ECLS command. Otherwise it must use the CLS2 command.

3.3.2.3 SENDING HOST'S RESPONSE TO CONTROL MESSAGES

(1). RSR command: the sender must transmit a SFS command to the receiver for the link involved.

(2). ECLS command: The sender must cease transmission, if it has not already done so, and issue an ECLS command if it has not already issued either a ECLS or CLS2 command.

(3) CLS2 command: The sender must compare the LRN and MSN values of the CLS2 command with its own values of the LRN and MSN for the connection involved. If an error is indicated, it may either close the connection with an ECLS, or initiate recovery action as specified in the section 3.3.2.1.

(4). LMR command for a connection (i.e., not a control link): The sender may follow any one of the following three courses of action:

1. Close the connection with an ECLS command.
2. Set the allocations for the link involved to zero, set LRN to that specified in the LMR command, and restart communication at the point of break.
3. Set the allocations for the link involved to zero, set the LRN to that specified in the LMR command, and send an LMS command to the receiver host informing him that one or more of the lost messages can not be retransmitted. After sending an LMS command, a sending host must not transmit any more messages on the link involved until and unless it receives an LMA command from the receiver host.

(Comments: As we have mentioned before (Section 2.3), the decision regarding which course of action to follow depends upon a number of factors. For example, if a host has implemented only the detection of lost messages aspect of our protocol (and no recovery), then it will chose the first option of closing the connection.)

(5). LMR for a control link: The sender may take one of the following two actions:

1. Set the LRN to that specified in the LMR command and begin retransmission of lost messages
2. Set the LRN to that specified by the LMR command, and insert a Reset command at the break point.

(Comment: If a sending host can not retransmit messages lost on a control link, then this protocol requires that all communication between the two host be broken, and reinitialized. We do not explicitly specify the actions that are associated with the exchange of Reset commands. These actions are specified by the Host-to-Host protocol.)

(6). LMA command: When a sending host receives an LMA command matching an LMS command previously issued by it, it may resume transmission.

(Comments: The protocol does not require the sending host to take any specific action with regard to a SFR. However, a sending host may use the information contained in the SFR command regarding the state of transmission. From a SFR command a sender host may determine what messages have been received properly. The sender may use this information to cleanup its buffer space or retransmit messages that it might suspect are lost.)

3.3.2.4 RECEIVING HOST'S RESPONSE TO CONTROL MESSAGES

(1). RSS command: A receiver is obligated to transmit a SFR to the sender for the link involved.

(2). ECLS command: The receiver must close the connection by issuing an ECLS command if it has not already done so.

(3). CLS2 command: A receiver must compare the LRN and MSN values of the command with its own values for the connection involved. If an error is indicated, it may either close the connection by an ECLS command or initiate recovery procedures as specified in section 3.3.2.2.

(4). LMS command: The receiver may take one of the following two courses of action:

(1). Close the connection specified by the LMS command, by issuing an ECLS command.
(2). Set the link involved to be prepared to receive messages starting with the sequence number MSN + COUNT, where MSN and COUNT are those specified by the LMS command. (Comment: This action implies that receiver is willing to accept the loss of messages specified by the LMS command.)

(Comments: The protocol does not require the receiver to take any specific action with regard to a SFS command. However a receiver host may use the information contained in it.)

4.0 CONCLUDING REMARKS

The design of this protocol has been governed by three major principles. First, we believe that to be useful within the ARPA Network, any new protocol must be compatible with the existing protocols, so that each host can make the transition to the new protocol at its own pace and without large investment. Secondly, the protocol should tie into the recovery mechanism of the IMP-to-Host Protocol. The price we pay for this is the small MSN field and a message oriented protocol rather than a byte stream oriented protocol. The third consideration has been flexibility. While this protocol guarantees detection of lost messages, the philosophy behind the recovery procedures is that a host should have several options, each option providing a different degree of sophistication. A host can implement a recovery procedure that is most suitable for its needs and the capabilities of its machine. Even though two hosts may have implemented different recovery procedures, they can communicate with each other in a compatible way. In a network of independent machines of widely varying capabilities and requirements, this seems to be the only way of implementing such a protocol. Even though this protocol provides a variety of options in a given error situation, the choice of a specific action must be consistent with the basic requirements of the communication path. For example, partial recovery is not acceptable during file transfers. We fully expect the File Transfer Protocol to specify that if an irrecoverable error occurs, the file transfer must be aborted.

RFC # 669
NIC # 31435

D. W. DODDS, BBN-TENEXA
4 DECEMBER, 1974

NOVEMBER, 1974, SURVEY OF NEW-PROTOCOL TELNET SERVERS

TWO MONTHS HAVE ELAPSED SINCE OUR LAST SURVEY, AND THE APPEARANCE OF ADDITIONAL NEW-PROTOCOL SERVERS HAS PROGRESSED AT THE USUAL SNAIL'S PACE. THE CHANGES IN THIS LIST ARE (WITH HOST NUMBERS IN OCTAL):

SRI-AI (102) NOW HAS A NEW-PROTOCOL SERVER;
SDC-LAB (10) IS BACK ON THE NET AND THE LIST;
SDC-CC (110) IS COMING ON THE NET BUT STATUS IS AS YET UNKNOWN;
USC-ISI (126) AND USC-ISIB (226) (FORMERLY ISI-DEVTENEX) NOW HAVE NEW-PROTOCOL SERVERS;
SDAC-44 (32) HAS BEEN REMOVED--NO LONGER CLASSED AS A SERVER HOST;
HAWAII-500 (344) IS COMING ON THE NET, STATUS PRESENTLY UNKNOWN;
LONDON (52) HAS BEEN ADDED;
BBN-TENEXD IS NOW HOST 162 (FORMERLY 205).

WHAT FOLLOWS IS AN UPDATE OF THE SUMMARY AND TABULATION THAT APPEARED IN RFC #702.* IS THERE LIGHT AT THE END OF THE TUNNEL?

TOTAL SERVER HOSTS	37	100%
NO NEW-PROT SERVER	19	51%
UNKNOWN STATUS (NEW HOSTS)	2	6%
TOTAL NEW-PROT IMPLEM.	16	43%
NEW-PROT ON SOCKET 27, OLD ON SOCKET 1 (2)	9	24%
NEW-PROT ON 1 AND 27 (3)	6	16%
NEW-PROT ON 1 ONLY (3)	1	3%

NOTES:

* ALL DATA IN THIS REPORT WERE GATHERED VIA A SURVEYING PROGRAM RUN AT VARIOUS TIMES, PLUS A FEW MANUAL CHECKS TO FILL OUT THE DATA. WHAT IS REPORTED HERE IS THE WAY THE VARIOUS SERVERS WORK AS SEEN BY THE NEW-PROTOCOL USER TELNET AT BBNA, AS OF 4 DEC. 1974.

(2) THESE ARE THE SITES WHOSE OPERATION IS 100% CORRECT ACCORDING TO ALL PROTOCOLS AND CONVENTIONS, AS I UNDERSTAND THEM.

(3) WE REALIZE THAT SOME OF THE SERVERS THAT APPEAR HERE AS NEW-PROTOCOL SERVERS ON SOCKET 1 ARE ACTUALLY SERVERS WHICH ATTEMPT TO COMMUNICATE WITH BOTH OLD- AND NEW-PROTOCOL USER TELNETS ACCORDING TO WHAT CONTROL SEQUENCES ARE RECEIVED.

. TABULATION OF SERVER STATUS FOR ALL SERVER SITES:

HOST NO.	HOST NAME	SOCKET 1	SOCKET 27	NEW-PROT. OPTIONS IMPLEMENTED (IF ANY)
101	UCLA-CCN	OLD	X	
201	UCLA-CCBS	OLD	X	
2	SRI-ARC	OLD	X	
102	SRI-AI	OLD	NEW	I1,3,6; 03
3	UCSB-MOD75	OLD	X	
4	UTAH-10	OLD	X	
105	BBN-TENEX	OLD	NEW	I1,3,6; 03
305	BBN-TENEXA	OLD	NEW	I1,3,6; 03
106	MIT-DMS	NEW	NEW	I1,3; 03
206	MIT-AI	OLD	X	
306	MIT-ML	OLD	X	
7	RAND-RCC	OLD	X	
10	SDC-LAB	OLD	X	
110	SDC-CC	?	?	
11	HARV-10	NEW	X	I1,3; 03
12	LL-67	OLD*	X	
112	LL-TX-2	OLD	X	
13	SU-AI	NEW*	NEW*	I1,3
15	CASE-10	OLD	X	
16	CMU-10B	NEW	NEW	I1,3; 03
116	CMU-10A	NEW	NEW	I1,3; 03
17	I4-TENEX	OLD	X	
217	KI4B-TENEX	OLD	X	
20	AMES-67	NEW	NEW	NONE
126	USC-ISI	OLD	NEW	I1,3,6; 03
226	USC-ISIB	OLD	NEW	I1,3,6; 03
27	USC-44	OLD	X	
327	USC-ECL	OLD	X	
37	CCA-TENEX	OLD	X	
40	PARC-MAXC	OLD	NEW	I1,3,6; 03
43	UCSD-CC	OLD	NEW	I0(!),3; 00,3
344	HAWAII-500	?	?	
52	LONDON	OLD*	X	
53	OFFICE-1	OLD	X	
54	MIT-MULTICS	NEW	NEW	NONE
61	BBN-TENEXB	OLD	NEW	I1,3,6; 03
162	BBN-TENEXD	OLD	NEW	I1,3,6; 03

KEY:

X	NO SERVER AT THIS SOCKET
?	STATUS NOT ASCERTAINED--UNABLE TO CONNECT TO HOST
I#	OPTION # IMPLEMENTED INCOMING TO USER (SERVER SAYS "WILL #")
0#	OPTION # IMPLEMENTED OUTGOING FROM USER (SERVER SAYS "DO #")

(# IS OPTION NUMBER IN NEW PROTOCOL. ALL OPTIONS IMPLEMENTED BY ANYONE ARE:

0	TRANSMIT-BINARY
1	ECHO
3	SUPPRESS-GO-AHEAD
6	TIMING-MARK)

NOTE: * THESE SERVERS RETURN IMPROPER RESPONSES TO SOME TELNET OPTION REQUESTS.